ORCA: Outlier-aware Rotational Cluster Factorization for Efficient Low-Rank LLM Compression

Han Cho
Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, USA
hcho@gatech.edu

Fernando Camacho
Laboratory for Physical Sciences
College Park, USA
fercamacho@lps.umd.edu

Saibal Mukhopadhyay
Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, USA
saibal.mukhopadhyay@ece.gatech.edu

***Abstract*—The immense size and computational cost of Large Language Models (LLMs) present significant barriers to their widespread deployment. Low-Rank Approximation (LRA) offers a promising, hardware-friendly solution by factorizing large weight matrices into more compact forms. A key insight is that the accuracy of this factorization can be significantly enhanced by first applying a geometric transformation to the model's weights. In this work, we introduce ORCA (Outlier-aware Rotational Cluster Factorization), a novel LRA framework that uses structured clustering-based factorization of weight matrices. We first apply an orthogonal transform to restructure the weight geometry to be more suitable for clustering. We then apply a group-wise clustering algorithm to the transformed weights to achieve a precise approximation. Furthermore, we demonstrate that this factorized representation enables an optional clustered attention reformulation, which reduces the algorithmic complexity of attention computation by performing attention computations directly in the compressed domain. Through experiments on the LLaMA and OPT model families, we show that ORCA can compress models by 75% while retaining over 96% of the original zero-shot accuracy on LLaMA2-13B, achieving a competitive compression-accuracy trade-off.**

***Index Terms*—neural network compression, weight clustering, attention acceleration, efficient inference**

I. Introduction

Large Language Models (LLMs) have become foundational across a vast area of applications, demonstrating remarkable performances in complex reasoning and generation tasks. However, this success is built on top of immense scale, with models containing billions of parameters that demand extensive computational and memory resources. The resulting costs create a significant barrier to their widespread deployment, making inference expensive and hindering their use in resource-constrained environments. Consequently, model compression has become a critical field of research aimed at making these powerful models more accessible and efficient.

While element-wise compression methods like scalar quantization [1]–[3] have been heavily explored, our work focuses on the structural path of Low-Rank Approximation (LRA) [4]–[7], specifically weight clustering to achieve a low-rank factorization of model weights, which has the advantage of capturing and preserving high-dimensional structure. This approach decomposes a weight matrix into a small set of learned centroids and corresponding matrix of integer indices, producing a hardware-friendly dense representation. However, the primary challenge for finding an accurate factorization is the presence of vector-wise outliers: a small number of out-of-distribution weight vectors that greatly influence the clustering process. These outliers stretch the geometric space, pulling cluster centroids away from dense regions and leading to a poor low-rank approximation for the majority of the in-distribution data. As illustrated in Fig. 1, weight vectors in large language models exhibit a small number of vector-wise outliers that distort clustering-based low-rank approximations

Notably, recent work leverages the principle of computational invariance, which allows for the application of orthogonal transformations to restructure a model's internal representations. This principle has been successfully leveraged to enable various compression goals, from reducing a model's embedding dimension [8] to improving quantization performance [9], [10]. However, its potential to improve the performance of clustering-based low-rank factorization remains a comparatively under-explored area.

In this work, we develop ORCA, a methodology to address vector-wise outliers that degrade the performance of low-rank approximation. We show that the principle of computational invariance can be applied for this purpose by applying a random Hadamard transform to reshape the high-dimensional geometry of the weight vectors. This transformation reduces the influence of high-leverage outliers, producing a distribution that is more amenable to structured clustering. As a result, the clustered weights yield a more faithful low-rank approximation and enables robust performance at high compression ratios.

Our main contributions are as follows:

1) We identify and quantify the effect of out-of-distribution outlier vectors in LLM weight matrices, showing they are a primary obstacle to effective low-rank approximation via weight clustering.
2) We propose a novel LRA framework that applies a random Hadamard transform to restructure the weight geometry, mitigating the influence of outliers and enabling structured clustering to factorize weight matrices effectively.

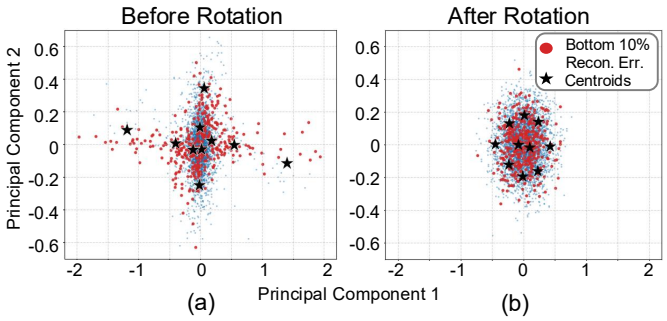

Fig. 1: Vector-wise outliers in weight clustering. (a) PCA projection of a representative weight matrix, highlighting out-of-distribution (OOD) vectors (red) that affect centroid placement. (b) Weight distribution after applying a random Hadamard transform, mitigating the effects of OOD vectors.

3) We show that our low-rank factorization approach is orthogonal to element-wise methods like quantization and can be combined to achieve further gains in efficiency and higher compression ratios.
4) We show that the proposed clustered weight representation enables an optional reformulation of the attention computation under matched query–key clustering, allowing parts of multi-head self-attention to be computed directly in the clustered domain.
5) We demonstrate through extensive experiments that ORCA achieves a state-of-the-art compression-accuracy trade-off. ORCA is able to compress models by up to 75% while retaining over 96% of the original zero-shot accuracy on LLaMA2-13B, outperforming existing LLM low-rank approximation works.

II. Background and Related Work

A. Low-Rank Approximation of LLM

Low-rank approximation (LRA) is a promising structural compression approach that aims to reduce model size by replacing large weight matrices with a more compact representation, often through factorization. Unlike element-wise methods that can produce sparse matrix representations, LRA-based methods typically result in smaller, dense matrices, which have high efficiency on modern hardware like GPUs. This approach has the advantage of capturing and preserving the high-dimensional structure of the weight matrices. The primary LRA techniques in recent literature can be categorized as follows:

SVD-based: A classic approach to LRA is through Singular Value Decomposition (SVD). Methods like SVD-LLM [4] and ASVD [7] factorize a weight matrix into singular vectors and values, and then achieve compression by truncating the components corresponding to the smallest singular values. While a standard baseline of LRA, the performance of the approximation can degrade significantly at higher compression ratios.

PCA-based: Another family of methods uses Principal Component Analysis (PCA) on the weights or activations directly to identify a low-rank subspace and operate within this more compact subspace. SliceGPT [8], for example, uses PCA to find the principal components of the hidden states and then "slices" off the dimensions corresponding to the least important components, effectively reducing the model's embedding dimension. Similarly, FLAT-LLM [5] uses a fine-grained, head-wise PCA to achieve a more targeted low-rank transformation.

While effective, these LRA methods operate on the fixed geometry of the pre-trained weights. However, this geometry may not be optimal for LRA, particularly due to the presence of outliers [2]. This raises a critical question: rather than approximating the existing weights, can we first transform them into a structure that is inherently more amenable to a low-rank approximation?

B. Computationally Invariant Transformations

A recent advance in LLM compression is the principle of computational invariance, which states that an orthogonal transformation can be applied within the network without changing the model's final output, provided it is properly reversed. This allows for the restructuring of weights and activations into a more compression-friendly format while preserving the model's function. This invariance is possible because an orthogonal matrix ($Q^TQ = QQ^T = I$) can be absorbed into adjacent weight matrices, enabling a powerful, lossless manipulation of the network's internal geometry.

Specifically, if the output weight matrix of one block is W_{out} and the input weight matrix of the subsequent block is W_{in}, the transformation is absorbed by modifying the weights as follows: the output matrix is post-multiplied by Q ($W_{out} \rightarrow W_{out}Q$), while the subsequent input matrix is pre-multiplied by its inverse, Q^T ($W_{in} \rightarrow Q^TW_{in}$). This ensures that the model's end-to-end computation remains unchanged ($A(W_{out}Q)(Q^TW_{in}) \rightarrow AW_{out}(QQ^T)W_{in} = AW_{out}W_{in}$). This principle has been successfully leveraged for various compression goals: SliceGPT [8] uses it for structured pruning, methods like QuaRot [9] and SpinQuant [10] use it to improve element-wise quantization, and QuIP# [11] applies it to enable vector quantization. Unlike prior rotation-based methods such as QuaRot and SpinQuant, which primarily mitigate scalar outliers for element-wise quantization, ORCA is motivated by vector-wise geometric distortion that destabilizes clustering-based low-rank approximation under high compression.

III. Proposed Method

A. Restructuring Weight Geometry with Orthogonal Transforms

In this work, we focus on orthogonal rotations instantiated using a random Hadamard transform (RHT). A primary challenge in applying low-rank approximation to large language models is the inherent geometry of their weight matrices. Although these weights exhibit substantial structural redundancy due to the structured computational graph of transformers (e.g., multi-head self-attention) [12], their distributions are often characterized by the presence of high-leverage outliers. As

shown in recent works [2], [13], such outliers stretch the geometric space, making the overall distribution difficult to compress.

Recent studies have demonstrated that orthogonal transformations can mitigate this issue. Methods such as QuaRot [9] and SpinQuant [10] apply rotations to address element-wise outliers in activation and weight distributions, producing more homogeneous, Gaussian-like scalar distributions that are well suited for low-bit quantization. In contrast, we leverage the same principle to reshape the vector-wise geometry of weight matrices, enabling more effective clustering-based low-rank factorization.

Our work builds on this principle but re-frames the outlier problem for a different compression paradigm: clustering-based low-rank approximation. For this approach, the primary challenge is not individual outlier values, but vector-wise outliers that distort the clustering process. As visualized through PCA in Fig. 1(a), the weight vectors are mostly organized in a dense clump, while a small number of out-of-distribution (OOD) vectors deviate significantly from this main distribution. Interestingly, many of these OOD vectors also fall within the bottom 10% when ranked by individual reconstruction error, suggesting that while they contribute little to reconstruction quality, they can still exert a disproportionate influence on the clustering geometry.

A key finding from our analysis is that a small number of OOD vectors disproportionately distort the clustering process by acting as gravitational centers for centroids. Although these vectors are few, they induce large reconstruction errors for the dense in-distribution region, which ultimately degrades model performance. Our goal is therefore to apply an orthogonal rotation that mitigates the influence of these outliers, allowing centroids to better represent the main data distribution.

To achieve this restructuring of the weight geometry for better clustering performance, we leverage a fixed RHT as shown in Fig. 1(b). This choice is motivated by several advantages. First, the fast Walsh-Hadamard transform allows the rotation to be computed efficiently in $O(n \log n)$ time, instead of $O(n^2)$ of a regular transform in dimension n. Second, as an orthogonal matrix, it is a norm-preserving operation that rotates the weight vectors without distorting the relative distances between vectors. Finally, it has been shown to be an effective method by creating a more incoherent, or homogeneous distribution. This restructuring through RHT is a pre-processing step that does not alter the model's output but significantly improves the performance of the subsequent clustering-based compression method.

B. Orthogonal Transform-based Clustering

Our proposed framework achieves a low-rank approximation of a given weight matrix through a three-stage process. This layer-wise method first regularizes the geometry of the weight distribution before learning a compact, clustered representation.

Stage 1: Geometric restructuring via Hadamard transform.

First, for a given weight matrix $W \in \mathbb{R}^{d_{in} \times d_{out}}$, we apply a fixed, RHT to regularize its structure. The transformed matrix is computed as $W_{rot} = WH$, where H is a RHT matrix of size $d_{out} \times d_{out}$. This preconditioning step mitigates the influence of high-magnitude outlier vectors, creating a more well-conditioned distribution for the subsequent clustering stage.

Stage 2: Low-rank approximations via group-wise clustering

The second stage performs the compression using a group-wise clustering algorithm, to split the weight matrix into smaller, more manageable parts. The objective is to independently cluster the columns within horizontal partitions of the weight matrix. First, the rotated weight matrix $W_{rot} \in \mathbb{R}^{d_{in} \times d_{out}}$ is partitioned along its row dimension. The d_{in} rows are divided into G groups, where each group contains k rows, such that $d_{in} = G \times k$, where the partition factor k is a hyperparameter. Smaller values of k allow for a more fine-grained clustering at the cost of increased storage overhead for the additional cluster indices. This creates G independent sub-matrices:

$$W_{rot} = \begin{bmatrix} W_{rot}^{(1)} & \cdots & W_{rot}^{(G)} \end{bmatrix}^\top \tag{1}$$

where each sub-matrix $W_{rot}^{(i)} \in \mathbb{R}^{k \times d_{out}}$ contains the i-th group of k rows.

We define our clustering process, $\phi(\cdot)$, as the function that maps each sub-matrix to its compressed representation. For each horizontal sub-matrix $W_{rot}^{(i)}$, the algorithm partitions its d_{out} columns (each of row dimension k) into a dedicated set of c centroids. This results in G independent sets of centroids, $C^{(i)} \in \mathbb{R}^{k \times c}$, and G corresponding index vectors, $I^{(i)} \in \mathbb{Z}^{d_{out}}$.

The reconstruction process, $\phi^{-1}(\cdot)$, then uses these components to form the approximated weight matrix $\hat{W_{rot}}$. This reconstruction achieves a low-rank factorization of the original sub-matrix, given by:

$$\hat{W_{rot}^{(i)}} = \phi^{-1}(\mathbf{C}^{(i)}, \mathbf{I}^{(i)}) = \mathbf{C}^{(i)} \cdot S^{(i)} \tag{2}$$

where $S^{(i)}$ is a selection matrix constructed from the index vector $\mathbf{I}^{(i)}$. Crucially, while Eq. 2 is represented as a matrix product, it is implemented as a high-efficiency lookup table; the indices in $\mathbf{I}^{(i)}$ serve as pointers to gather centroids from $\mathbf{C}^{(i)}$, bypassing the need for dense matrix multiplication and significantly reducing computational complexity. The effective rank of each reconstructed sub-matrix is bounded by the number of centroids c

The full approximated matrix is the vertical concatenation of these reconstructed sub-matrices. The primary benefit is a significant reduction in storage. The original matrix requires $d_{in} \times d_{out}$ values, while the compressed form before the reconstruction requires storing a total of $c \times d_{in}$ values for the centroids and $G \times d_{out} \times \lceil \log_2 c \rceil$ bits for the indices. For typical LLM configurations where the number of clusters $c/d_{out} < 0.5$ and the partition factor $k > 8$, the total storage

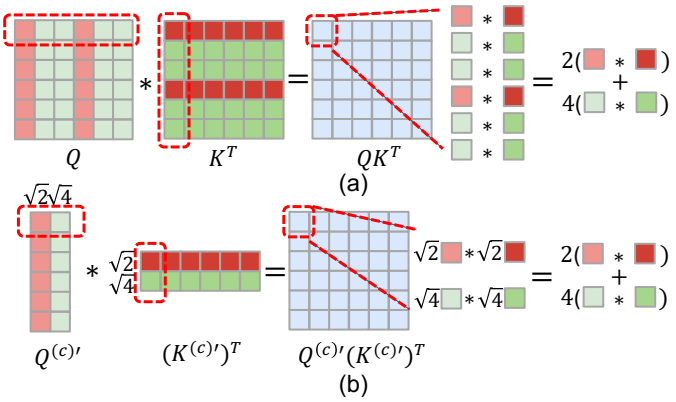

Fig. 2: Clustered attention mechanism. (a) Standard QK^T computation with matched QK clustering. (b) Equivalent reformulation using smaller, scaled activation matrices ($Q^{(c)'}$, $K^{(c)'}$).

cost of these indices is minimal, typically accounting for less than 2% of the original weight matrix size.

Stage 3: Data-aware centroid update. The final stage calibrates the centroids from Stage 2 using a modified GPTQ-style post-training procedure [1]. This step reduces residual reconstruction error using a small calibration set while preserving the clustering structure, serving as a refinement rather than the primary source of performance gains.

IV. Efficient Attention in the Clustered Domain

While the primary benefit of ORCA is improved low-rank approximation and memory reduction, the clustered weight representation also exposes additional structure that can be exploited to reformulate parts of the attention computation. In particular, when query and key projections share matched clustering assignments, the resulting activations admit a low-rank structure that allows the attention score computation to be expressed directly in the clustered domain.

In a standard multi-head self-attention layer, the input activations $X \in \mathbb{R}^{N \times d_{model}}$ are projected into Query (Q), Key (K), and Value (V) matrices using their respective weights. Because the query and key weights are clustered column-wise, the resulting activations inherit the same group structure, allowing Q and K to be expressed in terms of group-wise activation centroids. The scaled dot-product attention of a single head is then formulated as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_{head}}}\right)V. \tag{3}$$

The primary computational bottleneck in this operation is the QK^T matrix multiplication, which scales quadratically with the sequence length N. Our compressed representation avoids the need to compute this large matrix multiplication directly.

To alleviate this high computational requirement, we introduce a matched query-key clustering scheme. The core of this method is that for each attention head, the columns of the query weight matrix (W_Q) and the key weight matrix (W_K) are clustered into G clusters using the same cluster index assignments. Fig. 2 shows a sample illustration of the clustered attention through matched query-key clustering.

This "matching" imposes a specific low-rank structure on the resulting Query (Q) and Key (K) activation matrices. All columns within a given group become identical copies of a corresponding centroid vector. This allows the activation matrices to be factorized as:

$$Q = C^{(Q)}S \quad \text{and} \quad K = C^{(K)}S \tag{4}$$

where $C^{(Q)}, C^{(K)} \in \mathbb{R}^{N \times G}$ are the matrices containing the effective activation values for each of the G cluster groups. The matrix $S \in \mathbb{R}^{G \times d_{head}}$ is a selection matrix, determined by the shared column clustering, that maps these group activations to the original head dimension.

This shared structure allows for a significant simplification of the QK^T matrix multiplication. The score between the i-th query token and the j-th key token can be shown to be a weighted sum of the products of their group activation values:

$$QK^T_{i,j} = \sum_{g=1}^{G} |\mathcal{G}_g| \cdot C^{(Q)}_{i,g} \cdot C^{(K)}_{j,g}, \tag{5}$$

where $|\mathcal{G}_g|$ is the size (number of column vectors) of the g-th cluster group.

Crucially, this entire $N \times N$ matrix multiplication can be rewritten. The sum of weighted dot products is equivalent to a single, much smaller matrix multiplication involving only scaled centroids. We define the scaled centroid matrices as:

$$Q^{(c)'} = C^{(Q)} \cdot \text{diag}(\sqrt{|\mathcal{G}_1|}, \ldots, \sqrt{|\mathcal{G}_G|}) \in \mathbb{R}^{N \times G}, \tag{6}$$

where the same scaling is applied to the key centroids to obtain $K^{(c)'}$. Since this scaling is applied column-wise to the group activation matrices, it can be absorbed directly into the original weight matrices (W_Q, W_K) prior to test time without any change to the final computation.

The full attention score matrix can now be computed efficiently as[1]:

$$QK^T = Q^{(c)'}(K^{(c)'})^T. \tag{7}$$

This reduces the complexity of the score calculation from $O(N^2 d_{head})$ to $O(N^2 G)$, where the number of cluster groups G is significantly smaller than the head dimension d_{head}. This provides a direct and substantial reduction in algorithmic complexity without any look-up or gather operations.

V. Experiments

A. Setup

We implement ORCA using the QuaRot repository, which is built upon the HuggingFace [14] and PyTorch framework [15]. For the main results presented in Tables I, II, III, ORCA is configured with a partition factor $k = 16$. We use the K-means algorithm for clustering. The centroids are calibrated

[1]Architectures that apply non-linear transformations (e.g., Rotary Position Embeddings) to queries and keys prior to the dot product require a modified formulation.

TABLE I: Comparison of WikiText-2 Perplexity (ppl) at various compression ratios across OPT and LLaMA2 model families.

		OPT					LLaMA2	
Method	Ratio	125M	1.3B	2.7B	6.7B	13B	7B	13B
Dense	-	27.65	14.63	12.47	10.86	10.13	5.47	4.88
DISP-LLM	20%	39.87	21.70	17.07	14.06	-	9.84	7.11
SliceGPT	20%	34.26	16.43	13.73	11.48	10.66	6.64	5.81
ProcrustesGPT	20%	36.08	-	13.95	-	10.67	6.54	5.71
SVD-LLM	20%	-	-	-	-	-	7.84	7.37
ORCA	20%	**27.91**	**14.93**	**12.68**	**11.00**	**10.22**	**5.49**	**4.94**
ORCA	50%	31.19	16.11	12.93	11.12	10.29	5.56	4.99
ORCA	75%	32.44	16.79	13.32	11.32	10.45	5.71	5.04

TABLE II: Zero-shot results for LLaMA2-7B and 13B models. Baselines use 20% ratio; ORCA uses 75%.

Model	Method	%	ARC-c	ARC-e	HS.	PIQA	WG.	Avg.
7B	Dense	0	46.25	74.58	75.99	79.11	68.82	68.95
	SliceGPT	20	35.15	56.10	53.04	65.78	62.98	54.61
	SVD-LLM	20	35.84	68.31	58.57	71.49	65.27	59.90
	FLAT-LLM	20	38.65	64.44	64.72	72.20	65.82	61.17
	Procrustes	20	41.98	68.35	69.72	73.94	**67.40**	64.28
	ORCA	75	**42.32**	**70.21**	**72.72**	**76.88**	63.77	**65.18**
13B	Dense	0	49.23	77.53	79.36	80.52	72.30	71.79
	SliceGPT	20	39.51	62.92	56.98	67.25	67.64	58.86
	SVD-LLM	20	39.93	71.00	63.47	72.91	67.17	62.90
	FLAT-LLM	20	46.25	**75.59**	73.36	75.84	**72.06**	68.62
	Procrustes	20	44.97	73.19	73.43	77.58	70.48	67.93
	ORCA	75	**47.61**	73.91	**76.86**	**78.89**	68.82	**69.22**

using 128 samples of 2048 length from WikiText-2 [16] with a modified algorithm based on GPTQ [1]. We evaluate our work on single NVIDIA A100 and NVIDIA H200 GPUs across the OPT [17], LLaMA1 [18], and LLaMA2 [19] model families, covering both language generation and zero-shot evaluation tasks.

B. Accuracy results

Language Generation Tasks: We evaluate the performance of ORCA on the language generation task using the WikiText-2 dataset [16]. Table I shows a comparison of perplexity for the OPT and LLaMA2 model families against several state-of-the-art low-rank approximation (LRA) baselines.

At a 20% compression ratio, ORCA demonstrates exceptional performance. Across all models, it consistently and significantly outperforms other LRA methods like DISP-LLM [20], SliceGPT [8], SVD-LLM [4], and ProcrustesGPT [21]. Notably, for the LLaMA2 models, ORCA at 20% compression achieves a perplexity that remains comparable to that of the original dense model, which underscores the effectiveness of the approximation.

The primary advantage of our approach is its robustness at high compression ratios. This high compression rate is possible because the selection matrix that maps the centroids to the original column dimensions, a large and sparse matrix of zeros and ones, does not need to be stored explicitly. Instead, it

TABLE III: Evaluation of hybrid ORCA and GPTQ quantization. This approach remains stable in the 2-bit regime where standard GPTQ fails.

		LLaMA1		LLaMA2	
Method	Bits	7B	13B	7B	13B
Dense	16	5.68	5.09	5.47	4.88
GPTQ	4	6.10	5.36	6.09	5.16
ORCA (50%) + GPTQ (8b)	4.34	5.91	5.23	5.72	5.04
GPTQ	2	17920	4127	6064	1960
ORCA (50%) + GPTQ (4b)	2.34	12.57	10.81	11.52	10.20

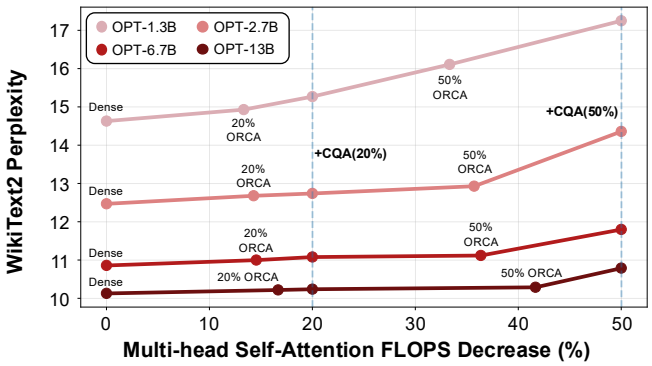

Fig. 3: Attention efficiency–accuracy trade-off for ORCA with clustered attention (CQA) across OPT models, measured relative to the dense attention baseline. Attention FLOPs include Q, K, V, QK^T, AV, and the output projection.

can be efficiently represented by a small vector of integer indices, where each index specifies which cluster group a column belongs to. As shown in the table, ORCA maintains very strong performance even at 50% and 75% compression, a regime where other LRA based methods are not typically evaluated or often degrade significantly. For instance, on OPT-6.7B, ORCA at 75% compression (11.32 ppl) outperforms the other baselines at 20% compression (14.06 ppl). This highlights that our geometric restructuring approach allows for high-ratio structural compression.

Zero-shot Tasks: We further evaluate ORCA's performance on a suite of five zero-shot commonsense reasoning tasks: ARC-challenge [22], ARC-easy [22], HellaSwag [25], PIQA [23], and WinoGrande [24]. Table II presents a comparison against several state-of-the-art LRA baselines including SliceGPT [8], SVD-LLM [4], FLAT-LLM [5], and ProcrustesGPT [21]. All baselines are evaluated at a 20% compression ratio, while ORCA is evaluated at a much higher 75% compression ratio.

For the LLaMA2-7B model, ORCA at 75% compression achieves an average accuracy of 65.18%, which not only outperforms all other LRA methods at 20% compression but is also higher than the next best baseline FLAT-LLM by nearly a full percentage point. This shows that our method compressed by an additional 55% still retains more zero-shot capability.

This trend continues with the larger LLaMA2-13B model.

ORCA achieves an average score of 69.22%. Compared to the dense model's performance, our 75% compressed LLaMA2-13B model retains over 96% of the original zero-shot accuracy, demonstrating that our method is highly effective at preserving the downstream task performance of LLMs even under high compression.

Joint Clustering & Quantization: To demonstrate that our structural compression method is orthogonal to and compatible with conventional element-wise compression techniques, we evaluate a hybrid approach that combines ORCA with a standard post-training quantization method, GPTQ [1]. Table III shows the results of applying GPTQ to our already-compressed models.

At approximately 4.34-bit precision (0.34 additional bits accounting for cluster indices), the combination of ORCA (50%) with 8-bit GPTQ significantly outperforms the standard 4-bit GPTQ baseline. For the LLaMA2-13B model, our hybrid approach achieves a perplexity of 5.04, which is an improvement over GPTQ's 5.16 showing the complementary benefits of the two methods.

The most significant advantage, however, is observed in the extreme low-bit regime. As shown in the table, standard 2-bit GPTQ fails completely, resulting in an unusable model with extremely high perplexity. In contrast, our hybrid approach of ORCA (50%) with 4-bit GPTQ, at a comparable effective bit of 2.34 bits, produces a stable and functional model with a reasonable perplexity.

Attention in the Clustered Domain: Fig. 3 evaluates the effect of applying optional clustered attention (CQA) on top of ORCA-compressed models. When matched query–key clustering is enabled, the attention computation can be reformulated in the clustered domain, resulting in a reduction in multi-head self-attention FLOPs. As shown in the figure, larger models are generally more robust to this approximation, exhibiting smaller increases in WikiText-2 perplexity for a given reduction in attention FLOPs, while smaller models show higher sensitivity. Overall, these results indicate that clustered attention provides an optional attention-level efficiency improvement that complements ORCA's primary compression gains.

VI. Conclusions

In this work, we introduce ORCA, a compression framework that combines geometric restructuring with clustering-based low-rank approximation. We show that pre-processing weights with a random Hadamard transform makes them more amenable to structured clustering, enabling robust performance at high compression ratios where prior structural methods often degrade. Across multiple model families, ORCA achieves a strong trade-off between model size and accuracy compared to existing low-rank approximation approaches. ORCA also provides two additional benefits. First, the proposed structural compression is orthogonal to element-wise methods, and combining ORCA with post-training quantization enables stable low-bit representations in regimes where standard quantization fails. Second, the clustered weight representation enables an optional reformulation of the attention computation, allowing parts of the multi-head self-attention to be executed in the clustered domain with reduced complexity. Overall, ORCA provides an effective framework for building more compact and efficient large language models.

References

[1] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "GPTQ: Accurate post-training quantization for generative pre-trained transformers," in *Proc. 11th Int. Conf. Learn. Represent. (ICLR)*, 2023.

[2] J. Lin et al., "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," in *Proc. Mach. Learn. Syst. (MLSys)*, pp. 87–100, 2024.

[3] Y. Zhao et al., "Atom: Low-bit quantization for efficient and accurate llm serving," in *Proc. Mach. Learn. Syst. (MLSys)*, pp. 196–209, 2024.

[4] X. Wang, Y. Zheng, Z. Wan, and M. Zhang, "SVD-LLM: Truncation-aware singular value decomposition for large language model compression," in *Proc. 13th Int. Conf. Learn. Represent. (ICLR)*, 2025.

[5] J. Tian, R. Solgi, J. Lu, H. Li, and Z. Zhang, "FLAT-LLM: Fine-grained low-rank activation space transformation for large language model compression," in *Findings of the EACL*, 2026.

[6] P. Chen, H.-F. Yu, I. Dhillon, and C.-J. Hsieh, "Drone: Data-aware low-rank compression for large nlp models," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2021.

[7] Z. Yuan et al., "Asvd: Activation-aware singular value decomposition for compressing large language models," *arXiv preprint arXiv:2312.05821*, 2023.

[8] S. Ashkboos, M. L. Croci, M. G. do Nascimento, T. Hoefler, and J. Hensman, "SliceGPT: Compress large language models by deleting rows and columns," in *Proc. 12th Int. Conf. Learn. Represent. (ICLR)*, 2024.

[9] S. Ashkboos et al., "QuaRot: Outlier-free 4-Bit inference in rotated llms," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2024.

[10] Z. Liu et al., "SpinQuant: LLM quantization with learned rotations," in *Proc. 13th Int. Conf. Learn. Represent. (ICLR)*, 2025.

[11] A. Tseng, J. Chee, Q. Sun, V. Kuleshov, and C. De Sa, "QuIP#: Even better llm quantization with Hadamard incoherence and lattice codebooks," in *Proc. 41st Int. Conf. Mach. Learn. (ICML)*, 2024.

[12] A. Vaswani et al., "Attention is all you need," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[13] G. Xiao et al., "Smoothquant: Accurate and efficient post-training quantization for large language models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, pp. 38087–38099, 2023.

[14] T. Wolf et al., "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[15] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019.

[16] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017.

[17] S. Zhang et al., "OPT: Open pre-trained transformer language models," *arXiv preprint arXiv:2205.01068*, 2022.

[18] H. Touvron et al., "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[19] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[20] S. Gao et al., "DISP-LLM: Dimension-independent structural pruning for large language models," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2024.

[21] E. Grishna, M. Gorbunov, and M. Rakhuba, "ProcrustesGPT: Compressing llms with structured matrices and orthogonal transformations," in *Findings of the ACL*, 2025.

[22] P. Clark et al., "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018.

[23] Y. Bisk, R. Zellers, J. Gao, R. L. Bras, and Y. Choi, "Piqa: Reasoning about physical commonsense in natural language," in *Proc. AAAI Conf. Artif. Intell.*, 2020.

[24] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, "Winogrande: An adversarial winograd schema challenge at scale," *Commun. ACM*, vol. 64, no. 9, pp. 99–106, 2021.

[25] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?," *arXiv preprint arXiv:1905.07830*, 2019.